

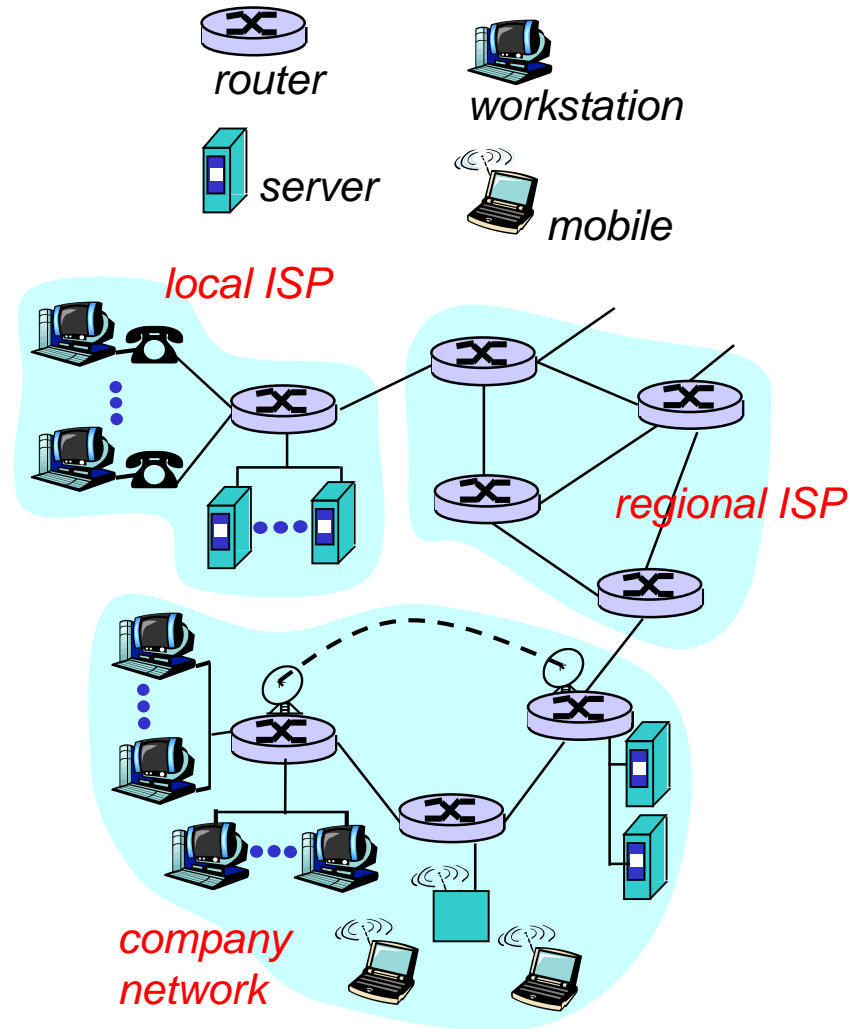
# Data Link Layer

Lecturers: Prof. Xiaoming Fu, Dr. Yali Yuan

Assistant: Yachao Shao, MSc

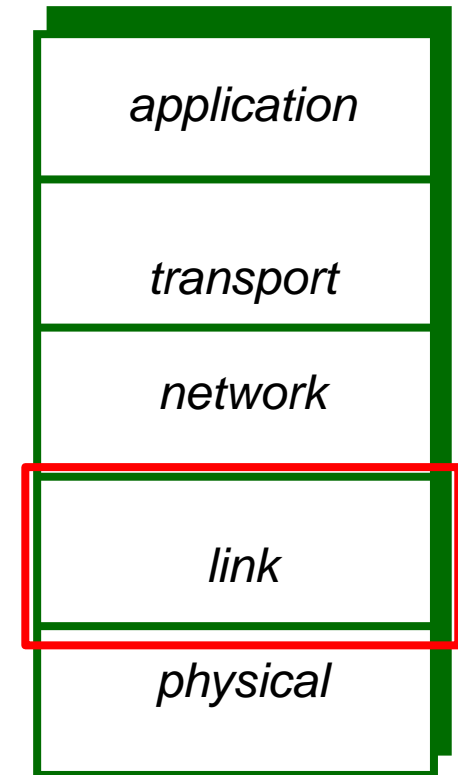
# What's the internet? A close look...

- millions of connected computing devices: hosts, end-systems
  - PCs, workstations, servers
  - PDAs, phones, toasters
  - running network apps
- communication links
  - fiber, copper, coax, radio, satellite
  - transmission rate = **bandwidth**
- routers: forward packets (chunks of data)

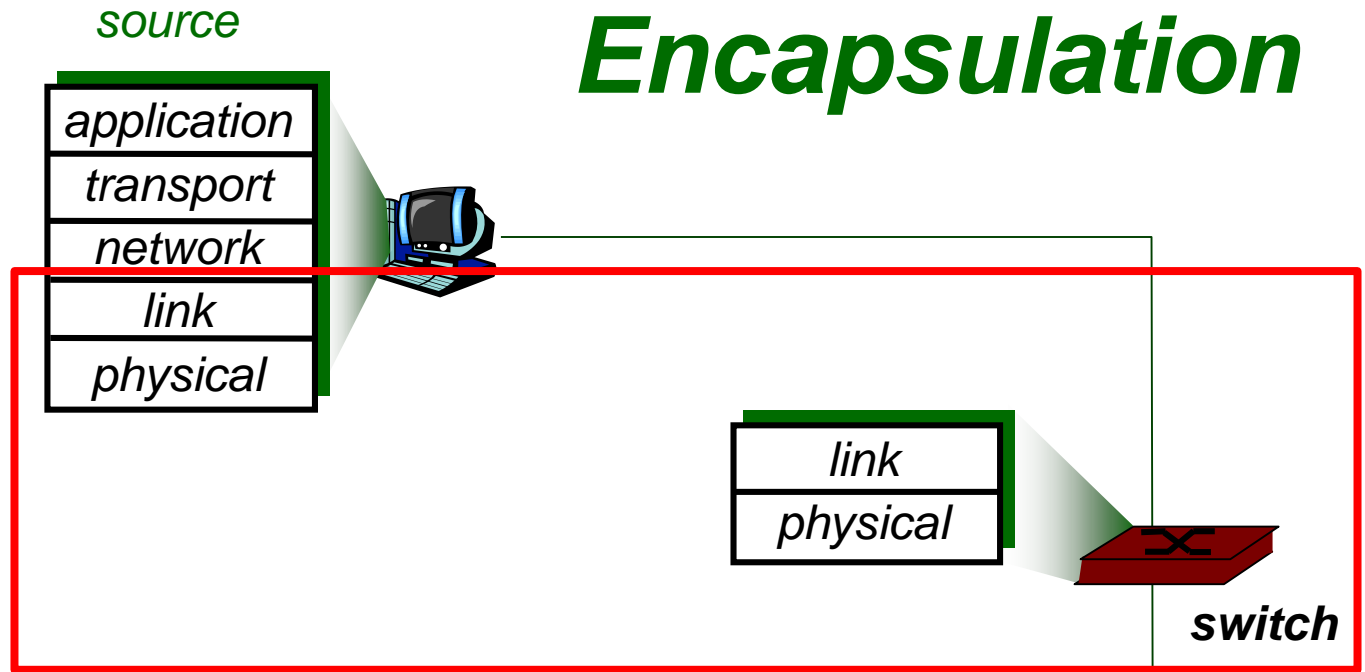
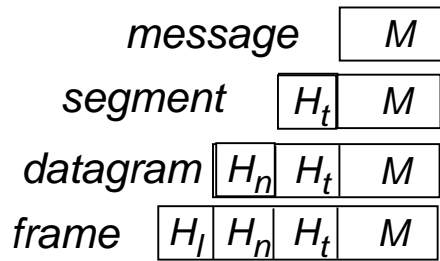


# Internet protocol stack

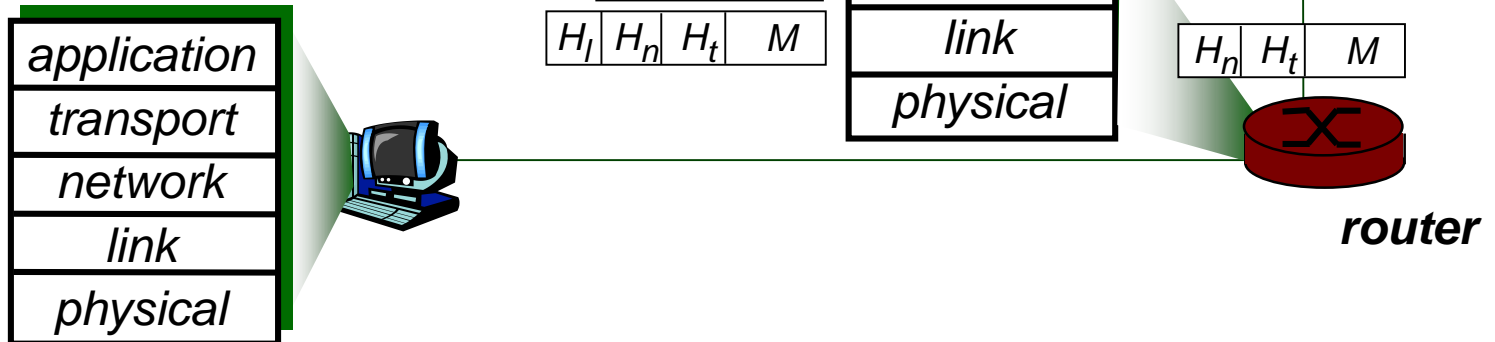
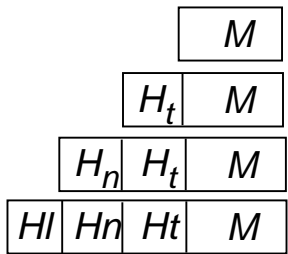
- **application:** supporting network applications
  - FTP, SMTP, HTTP
- **transport:** process-process data transfer
  - TCP, UDP
- **network:** routing of datagrams from source to destination
  - IP, routing protocols
- **link:** data transfer between neighboring network elements
  - PPP, Ethernet
- **physical:** bits “on the wire”



# Encapsulation



**destination**



# Chapter 2: The Data Link Layer

- Our goals:
- understand principles behind data link layer services:
  - reliable transmission of data over a link
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
- instantiation and implementation of various link layer technologies

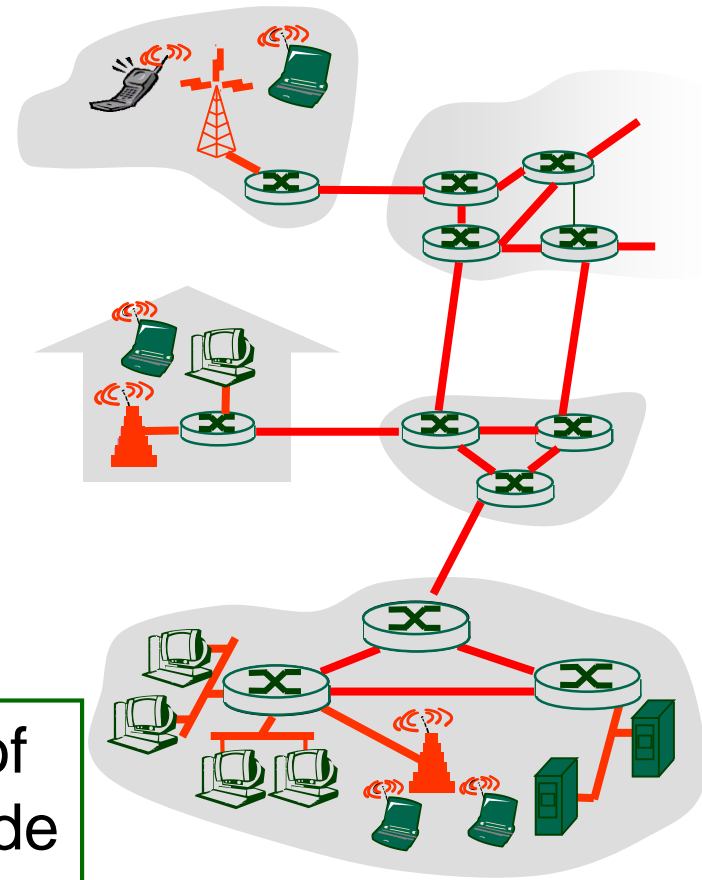
# Link Layer

- 2.1 Introduction and services
- 2.2 Error detection and correction
- 2.3 Multiple access protocols
- 2.4 Link-layer Addressing
- 2.5 Ethernet
- 2.6 Link-layer switches
- 2.7 PPP
- 2.8 Link virtualization: ATM, MPLS

# Link Layer: Introduction

## Some terminology:

- hosts and routers are **nodes**
- communication channels that connect adjacent nodes along communication path are **links**
  - wired links
  - wireless links
  - LANs
- layer-2 packet is a **frame**, encapsulates datagram



**data-link layer** has responsibility of transferring datagram from one node to adjacent node over a link

# Link layer: context

- datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- each link protocol provides different services
  - e.g., may or may not provide reliable data transfer (rdt) over link

## transportation analogy

- trip from Göttingen to Princeton
  - train: Göttingen -> FRA
  - plane: FRA -> NYC
  - limo: NYC -> Princeton
- tourist = **datagram**
- transport segment = **communication link**
- transportation mode = **link layer protocol**
- travel agent = **routing algorithm**



# Link Layer Services

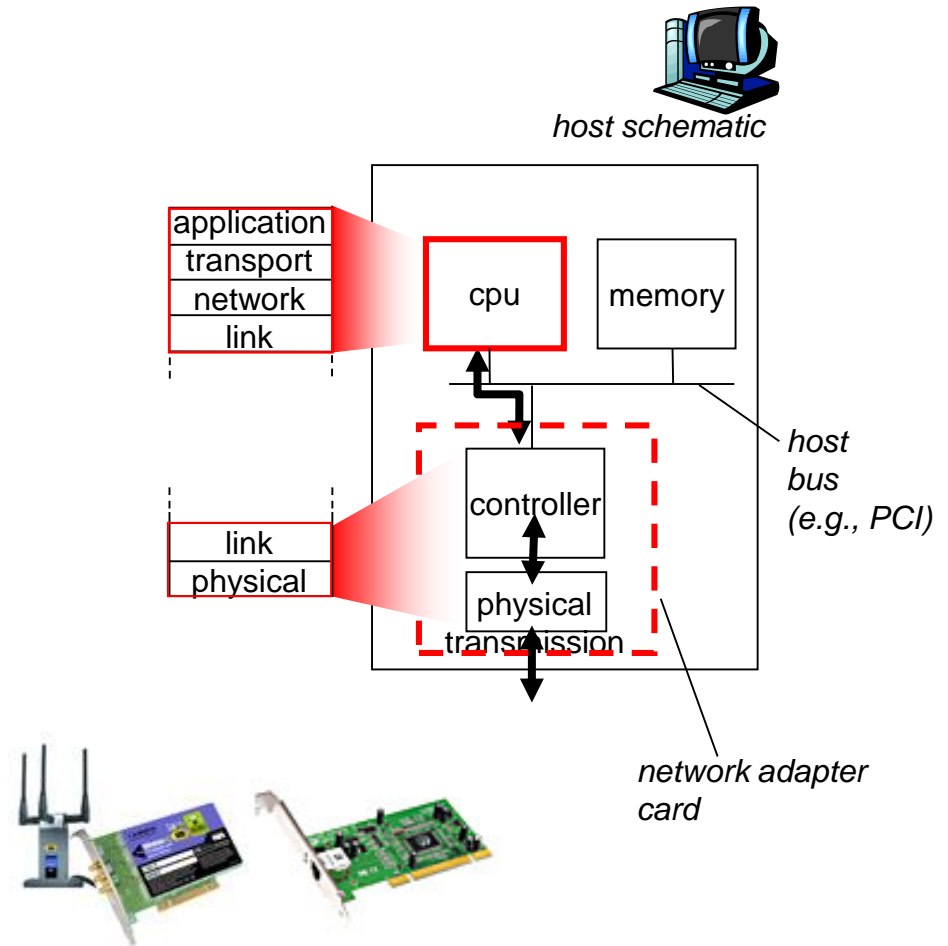
- *1. framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses used in frame headers to identify source and destination
    - different from IP address!
- *2. reliable delivery between adjacent nodes*
  - we will cover these so called Automatic repeat request (ARQ) algorithms in detail later
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates

# Link Layer Services (more)

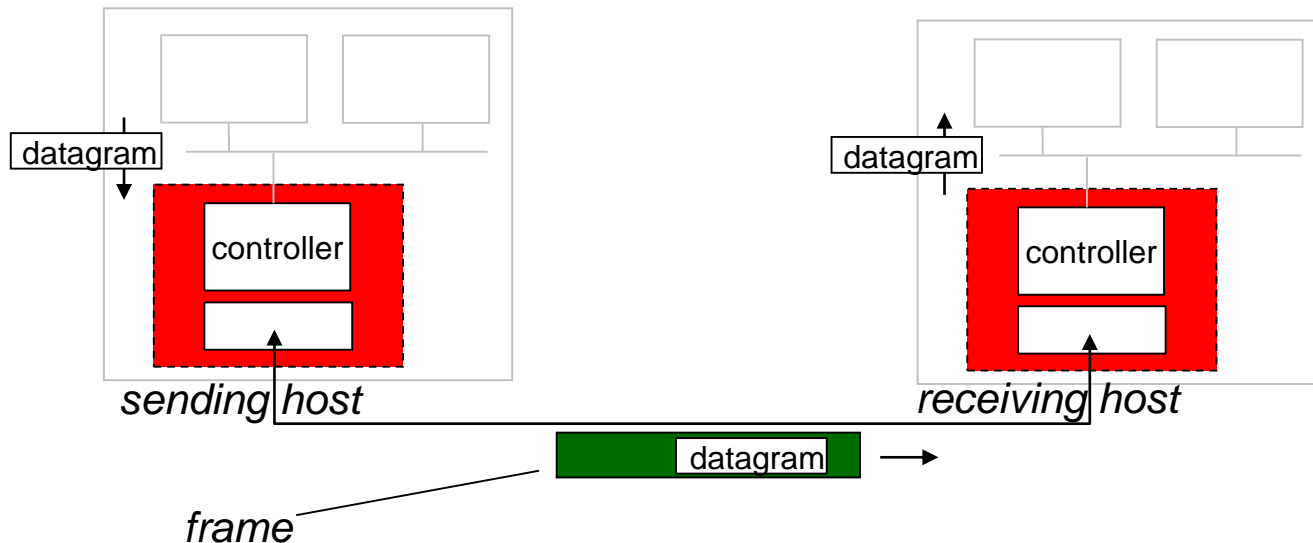
- *3. flow control:*
  - pacing between adjacent sending and receiving nodes
- *4. error detection:*
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *5. error correction:*
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *6. half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

# Where is the link layer implemented?

- in each and every host
- link layer implemented in “adaptor” (aka *network interface card* NIC)
  - Ethernet card, PCMCIA card, 802.11 card
  - implements link, physical layer
- attaches into host’s system buses
- combination of hardware, software, firmware



# Adaptors Communicating



- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

# Link Layer

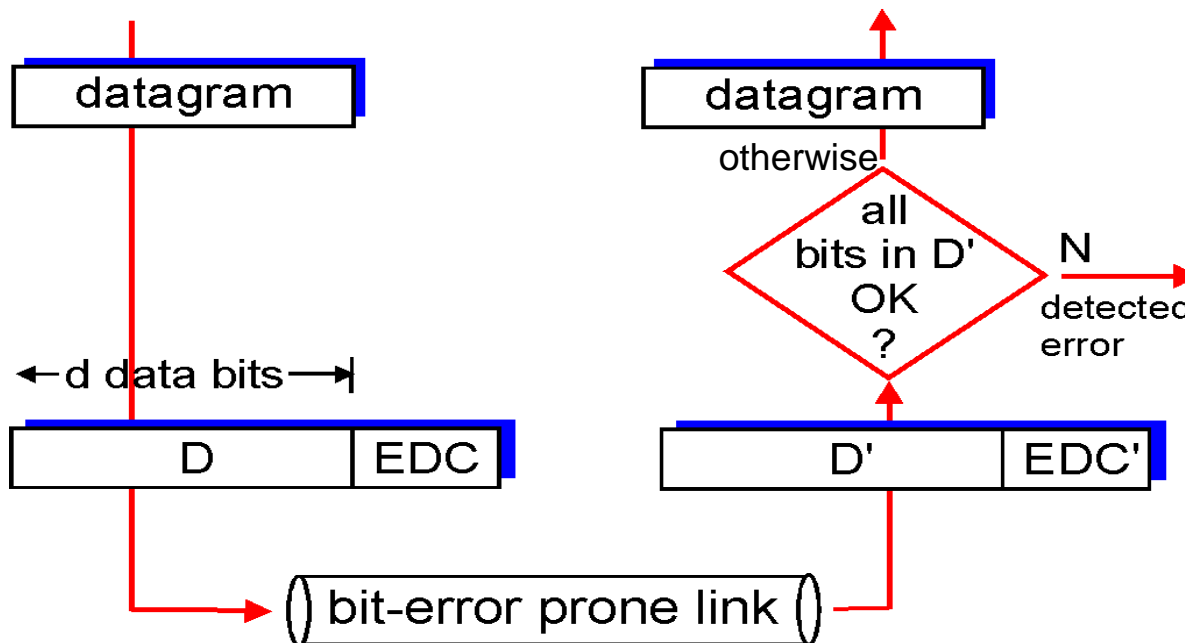
- 2.1 Introduction and services
- 2.2 Error detection and correction
- 2.3 Multiple access protocols
- 2.4 Link-layer Addressing
- 2.5 Ethernet
- 2.6 Link-layer switches
- 2.7 PPP
- 2.8 Link Virtualization: ATM. MPLS

# Error Detection

EDC= Error Detection and Correction bits (redundancy)

D = Data protected by error checking, may include header fields

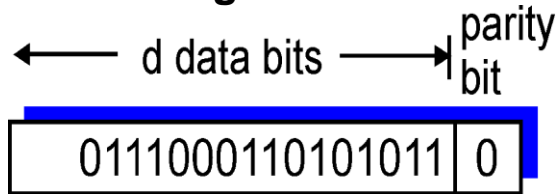
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



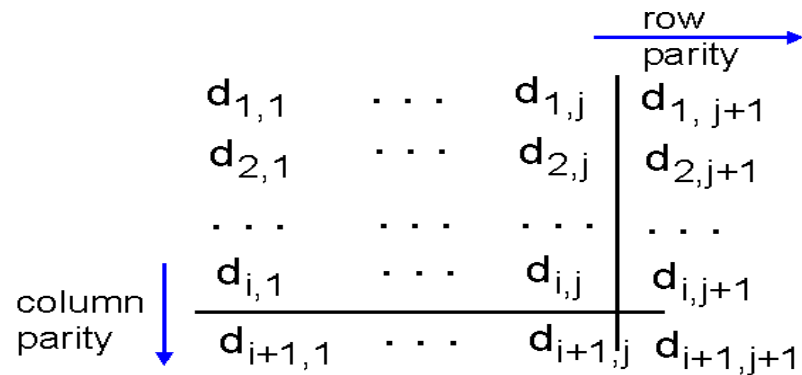
# Parity Checking

## Single (Odd) Bit Parity:

Detect single bit errors



## Two Dimensional (Even) Bit Parity: Detect *and correct* single bit errors



1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

*no errors*

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

parity error

parity error

*correctable  
single bit error*

# Internet checksum

Goal: detect “errors” (e.g., flipped bits) in transmitted packet (note: **used at transport layer only**)

## Sender:

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1’s complement sum) of segment contents
- sender puts checksum value into UDP checksum field

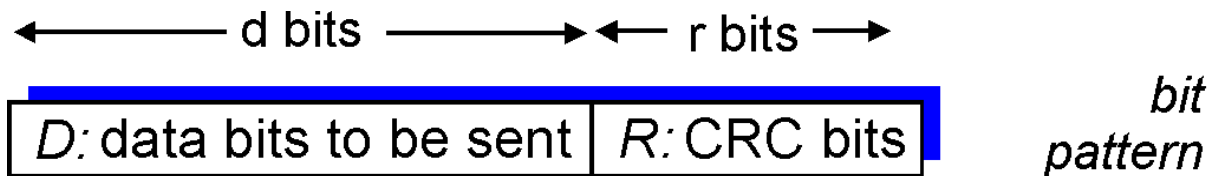
## Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected.  
*However, some 2-bit errors undetected...*



# Checksumming: Cyclic Redundancy Check

- view data bits, **D**, as a binary number
- choose  $r+1$  bit pattern (generator), **G**
- goal: choose  $r$  CRC bits, **R**, such that
  - $\langle D, R \rangle$  exactly divisible by  $G$  (modulo 2)
  - receiver knows  $G$ , divides  $\langle D, R \rangle$  by  $G$ . If non-zero remainder: error detected!
  - can detect all burst errors less than  $r+1$  bits
- widely used in practice (Ethernet, 802.11 Wi-Fi, ATM)



$$D * 2^r \text{ XOR } R$$

*mathematical formula*

# CRC Example

Want:

$$D \cdot 2^r \text{ XOR } R = nG$$

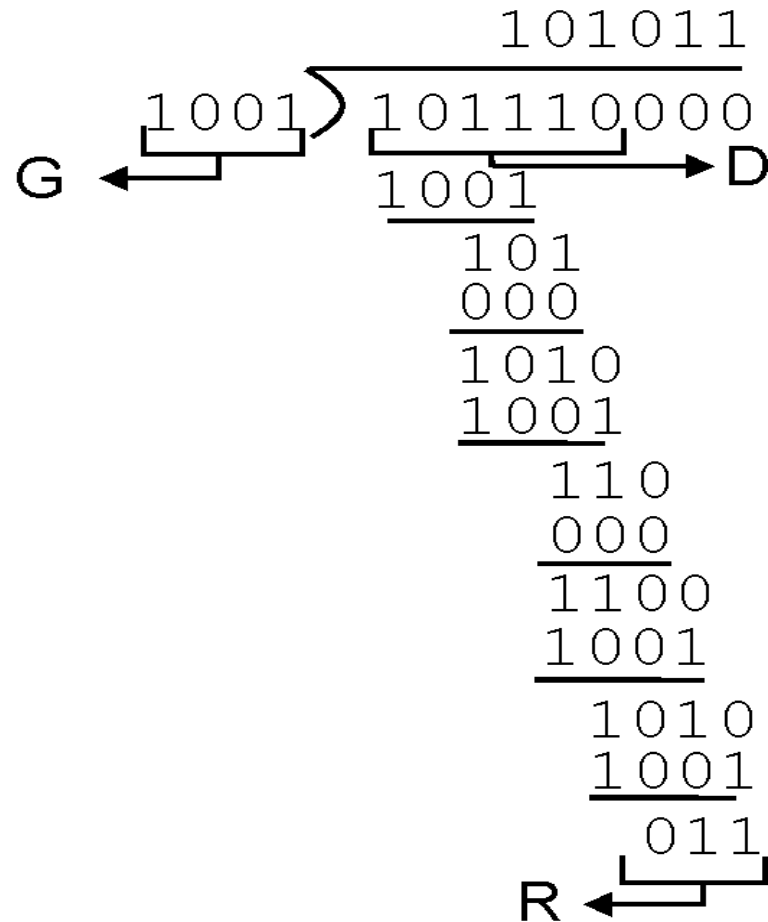
*equivalently:*

$$D \cdot 2^r = nG \text{ XOR } R$$

*equivalently:*

if we divide  $D \cdot 2^r$  by  $G$ ,  
want remainder  $R$

$$R = \text{remainder} \left[ \frac{D \cdot 2^r}{G} \right]$$



0x04C11DB7 is the CRC-32 polynomial

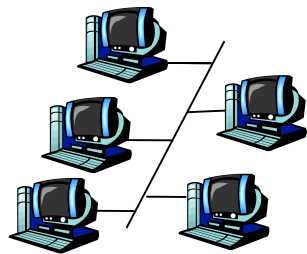
# Link Layer

- 2.1 Introduction and services
- 2.2 Error detection and correction
- 2.3 Multiple access protocols
- 2.4 Link-layer Addressing
- 2.5 Ethernet
- 2.6 Link-layer switches
- 2.7 PPP
- 2.8 Link Virtualization: ATM, MPLS

# Multiple Access Links and Protocols

Two types of “links”:

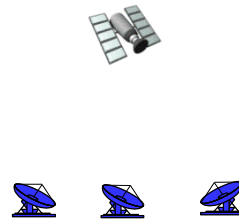
- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch and host
- **broadcast** (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN



shared wire (e.g.,  
cabled Ethernet)



shared RF  
(e.g., 802.11 WiFi)



shared RF  
(satellite)



humans at a  
cocktail party  
(shared air, acoustical)

# Multiple Access protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes:  
interference
  - **collision** if node receives two or more signals at the same time

## multiple access protocol

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination

# Ideal Multiple Access Protocol

## Broadcast channel of rate $R$ bps

1. when one node wants to transmit, it can send at rate  $R$ .
2. when  $M$  nodes want to transmit, each can send at average rate  $R/M$
3. fully decentralized:
  - no special node to coordinate transmissions
  - no synchronization of clocks, slots
4. simple

# MAC Protocols: a taxonomy

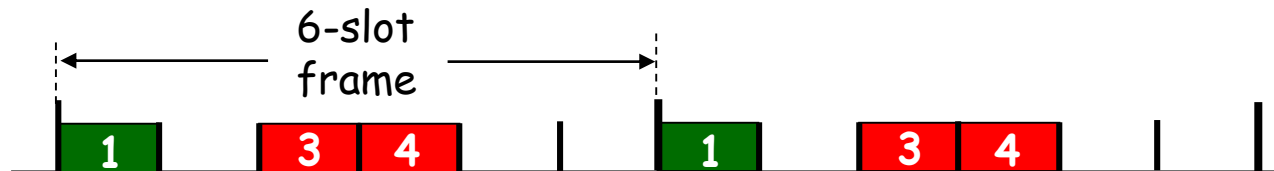
Three broad classes:

- Channel Partitioning
  - divide channel into smaller “pieces” (time slots, frequency, code)
  - allocate piece to node for exclusive use
- Random Access
  - channel not divided, allow collisions
  - “recover” from collisions
- “Taking turns”
  - nodes take turns, but nodes with more to send can take longer turns

# Channel Partitioning MAC protocols: TDMA

## TDMA: time division multiple access

- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle

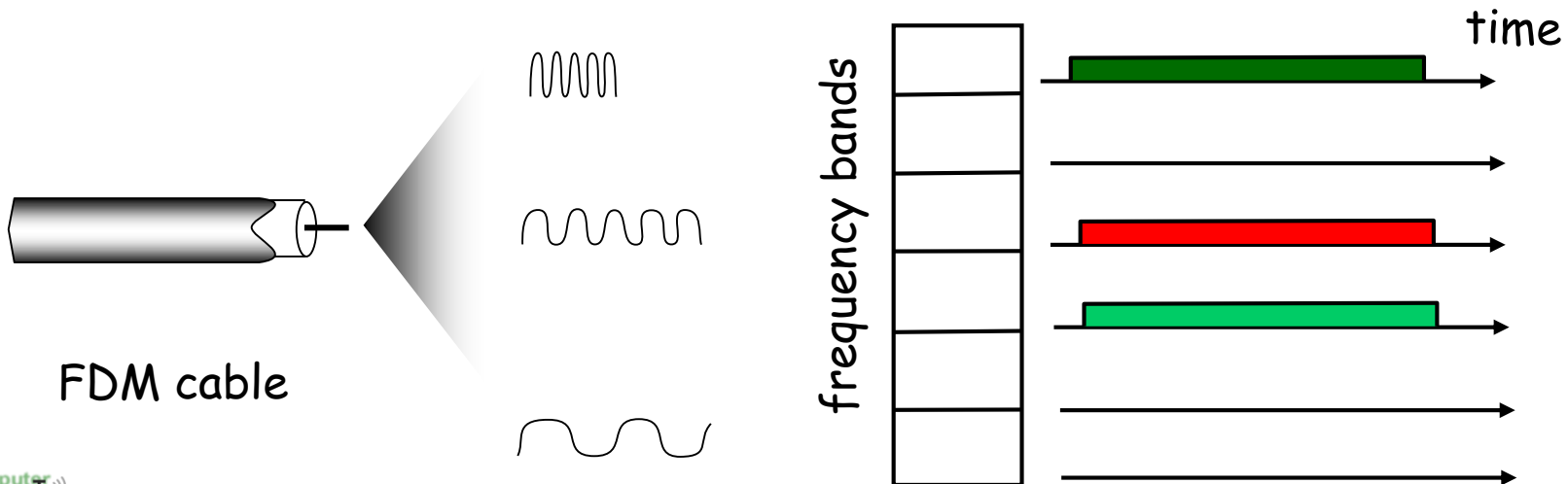




# Channel Partitioning MAC protocols: FDMA

## FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



# Random Access Protocols

- When node has packet to send
  - transmit at full channel data rate  $R$ .
  - no *a priori* coordination among nodes
- two or more transmitting nodes → “collision”,
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- Examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

# Slotted ALOHA

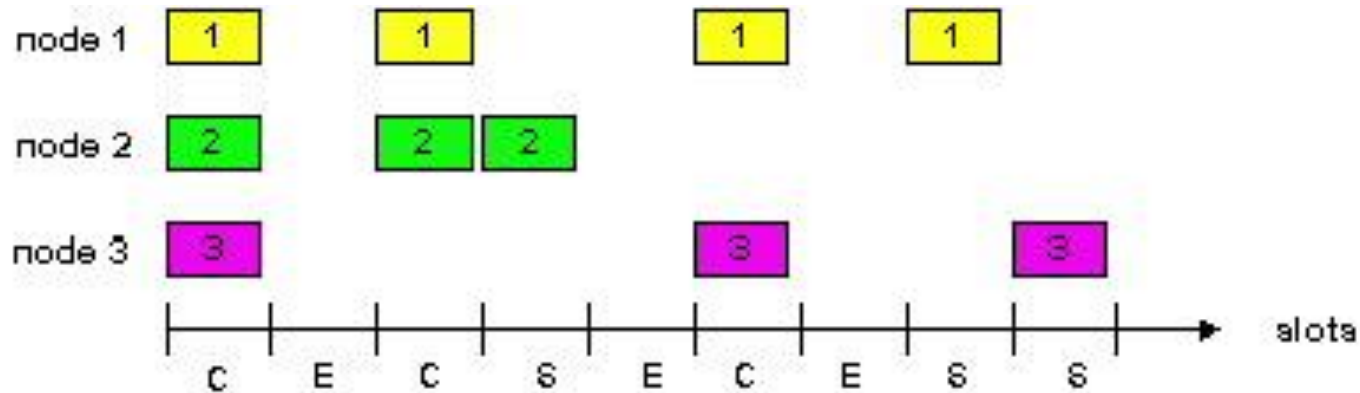
## Assumptions:

- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only at slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

## Operation:

- when node obtains fresh frame, transmits in next slot
  - *if no collision*: node can send new frame in next slot
  - *if collision*: node retransmits frame in each subsequent slot with prob.  $p$  until success

# Slotted ALOHA



## Pros:

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

## Cons:

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

# Slotted Aloha efficiency

**Efficiency** : long-run fraction of successful slots (many nodes, all with many frames to send)

- *suppose*:  $N$  nodes with many frames to send, each transmits in slot with probability  $p$
- prob that given node has success in a slot =  $p(1-p)^{N-1}$
- prob that *any* node has a success =  $Np(1-p)^{N-1}$

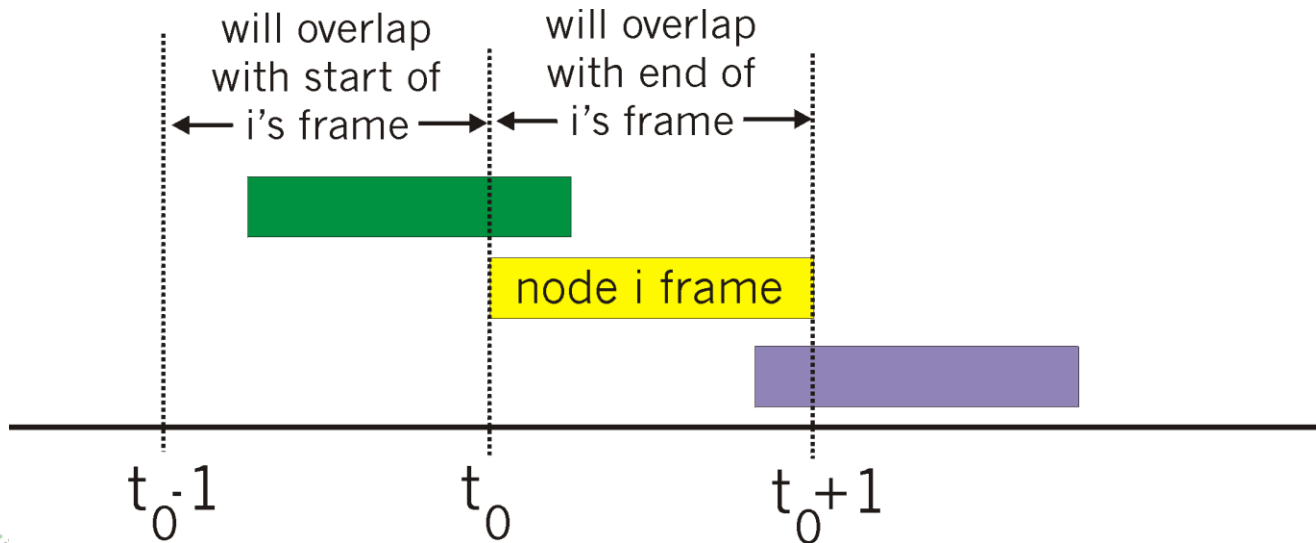
- max efficiency: find  $p^*$  that maximizes  $Np(1-p)^{N-1}$
- for many nodes, take limit of  $Np^*(1-p^*)^{N-1}$  as  $N$  goes to infinity, gives:  
Max efficiency =  $1/e = .37$

*At best*: channel used for useful transmissions 37% of time!



# Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - transmit immediately
- collision probability increases:
  - frame sent at  $t_0$  collides with other frames sent in  $[t_0-1, t_0+1]$



# Pure Aloha efficiency

$P(\text{success by given node}) = P(\text{node transmits}) \cdot$

$P(\text{no other node transmits in } [p_0-1, p_0] \cdot$

$P(\text{no other node transmits in } [p_0+1, p_0]$

$$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$$

$$= p \cdot (1-p)^{2(N-1)}$$

... choosing optimum  $p$  and then letting  $n \rightarrow \infty$  ...

$$= 1/(2e) = .18$$

*even worse than slotted Aloha!*

# CSMA (Carrier Sense Multiple Access)

CSMA: listen before transmit:

If channel sensed idle: transmit entire frame

- If channel sensed busy, defer transmission
  
- human analogy: don't interrupt others!



# CSMA collisions

collisions *can* still occur:

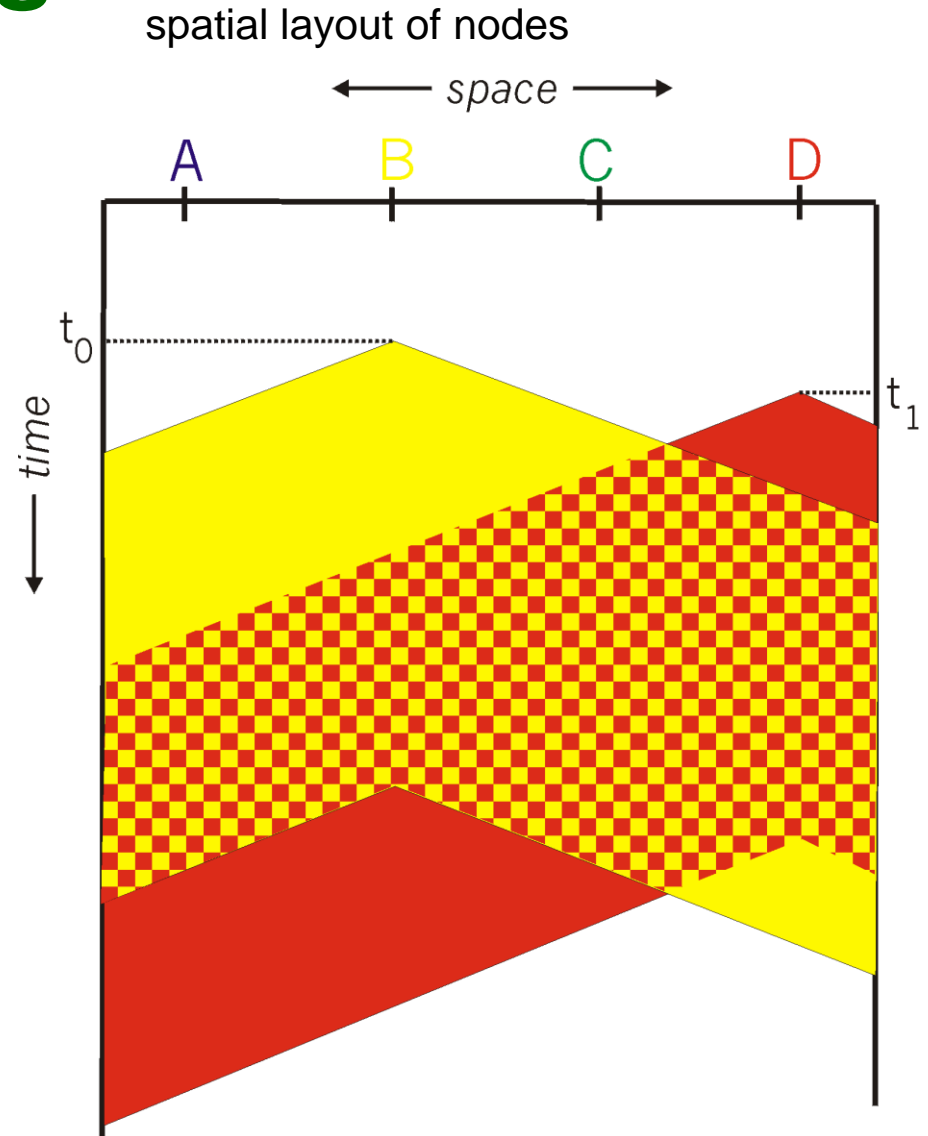
propagation delay means  
two nodes may not hear  
each other's transmission

collision:

entire packet transmission  
time wasted

note:

role of distance & propagation  
delay in determining collision  
probability

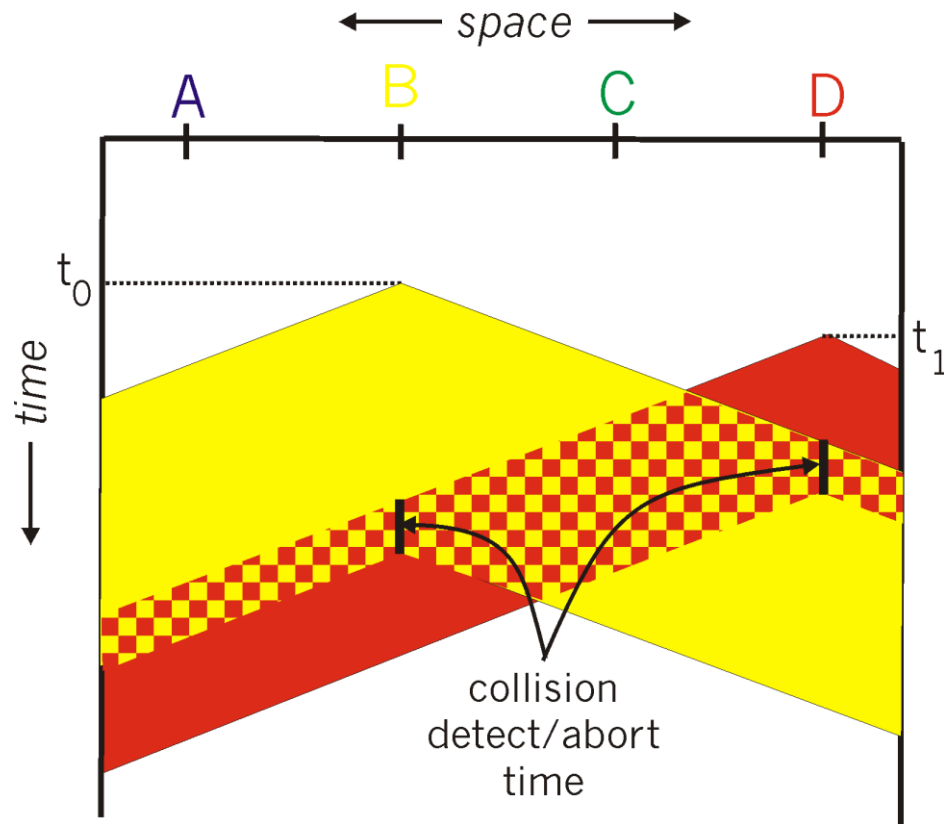


# CSMA/CD (Collision Detection)

**CSMA/CD:** carrier sensing, deferral as in CSMA

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage
- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength (e.g., half-duplex)
- human analogy: the polite conversationalist

# CSMA/CD collision detection



# “Taking Turns” MAC protocols

## channel partitioning MAC protocols:

- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access,  $1/N$  bandwidth allocated even if only 1 active node!

## Random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

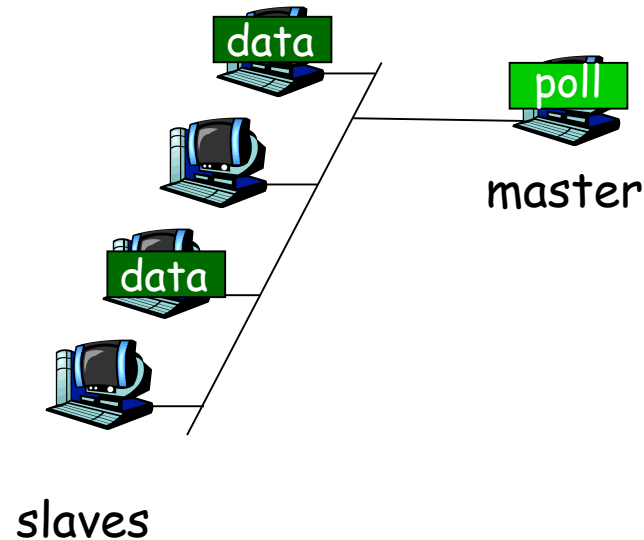
## “taking turns” protocols

look for best of both worlds!

# “Taking Turns” MAC protocols

## Polling:

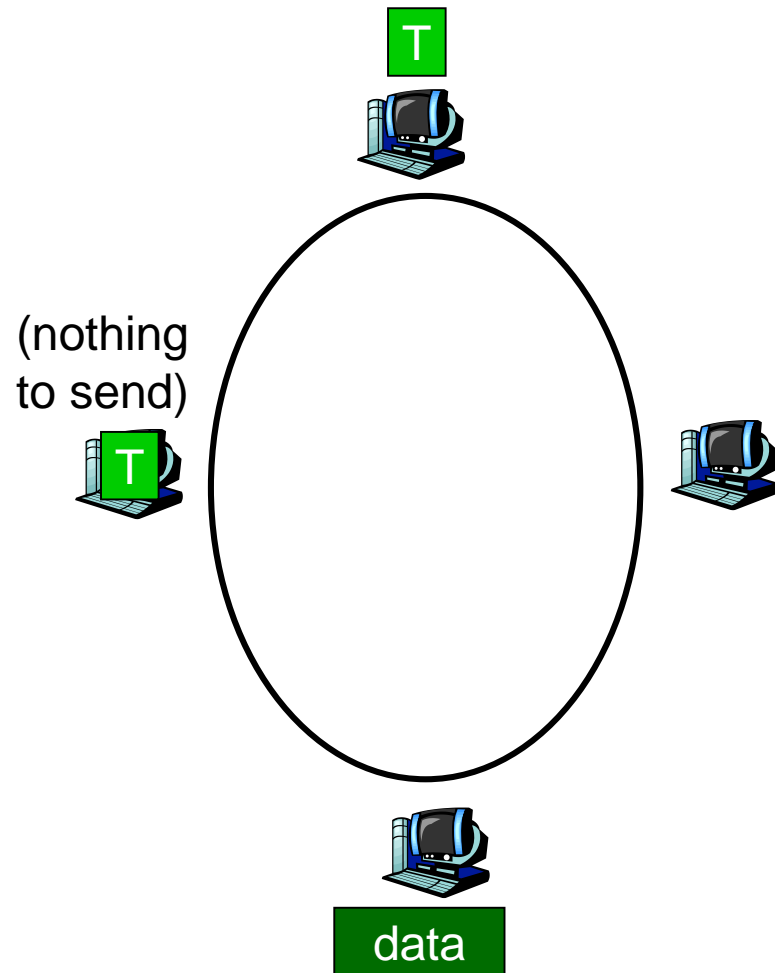
- master node “invites” slave nodes to transmit in turn
- typically used with “dumb” slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



# “Taking Turns” MAC protocols

## Token passing:

- control **token** passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)



# Summary of MAC protocols

- *channel partitioning*, by time, frequency or code
  - Time Division, Frequency Division
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11 (next week)
- *taking turns*
  - polling from central site, token passing
  - Bluetooth, FDDI, IBM Token Ring

# Link Layer

- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3 Multiple access protocols
- 5.4 Link-Layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches
- 5.7 PPP
- 5.8 Link Virtualization: ATM, MPLS

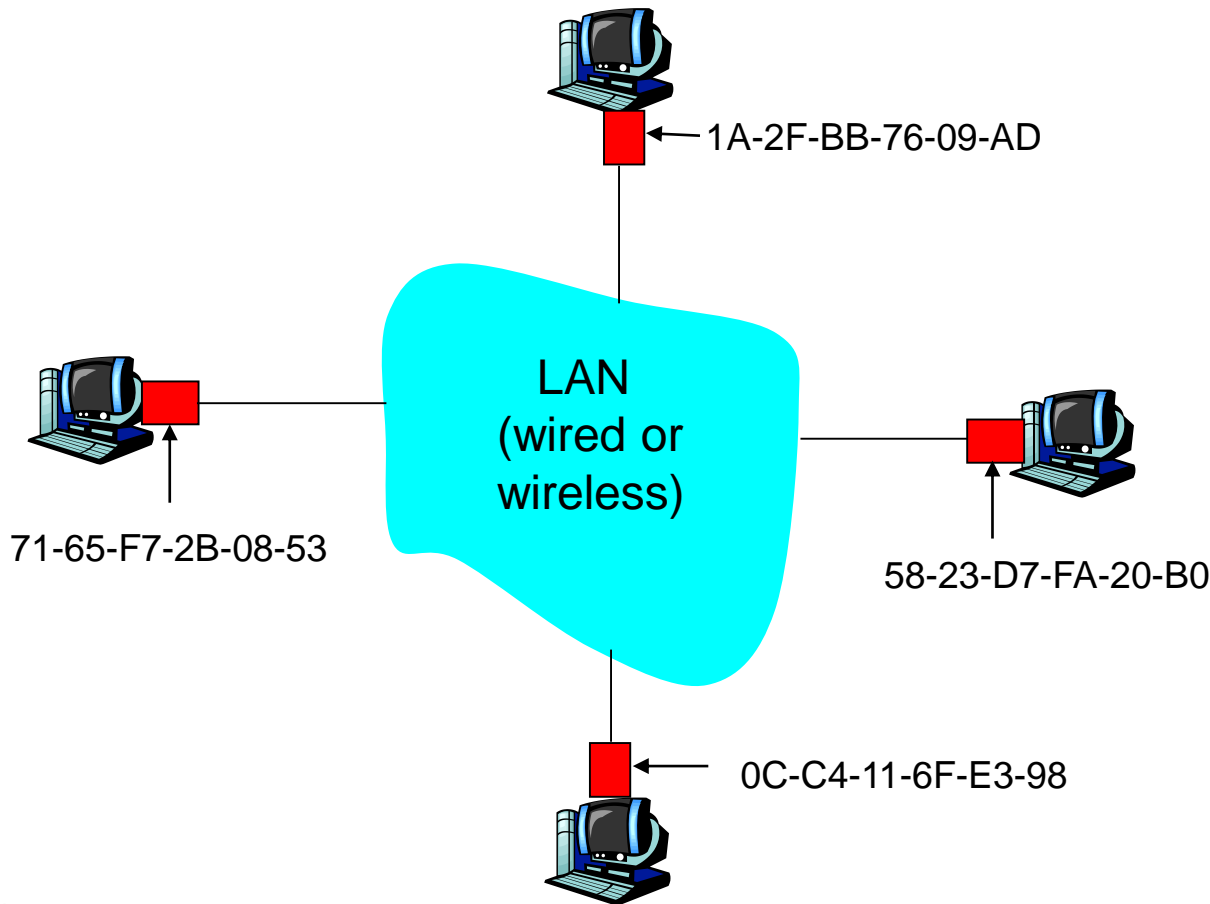


# MAC Addresses and ARP

- 32-bit IP address:
  - *network-layer* address
  - used to get datagram to destination IP subnet
- MAC (or LAN or physical or Ethernet) address:
  - function: *get frame from one interface to another physically-connected interface (same network)*
  - 48 bit MAC address (for most LANs)
    - burned in NIC ROM, also sometimes software settable

# LAN Addresses and ARP

Each adapter on LAN has unique LAN address



Broadcast address =  
FF-FF-FF-FF-FF-FF

■ = adapter

# LAN Address (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - (a) MAC address: like Social Security Number
  - (b) IP address: like postal address
- MAC flat address → portability
  - can move LAN card from one LAN to another
- IP hierarchical address NOT portable
  - address depends on IP subnet to which node is attached

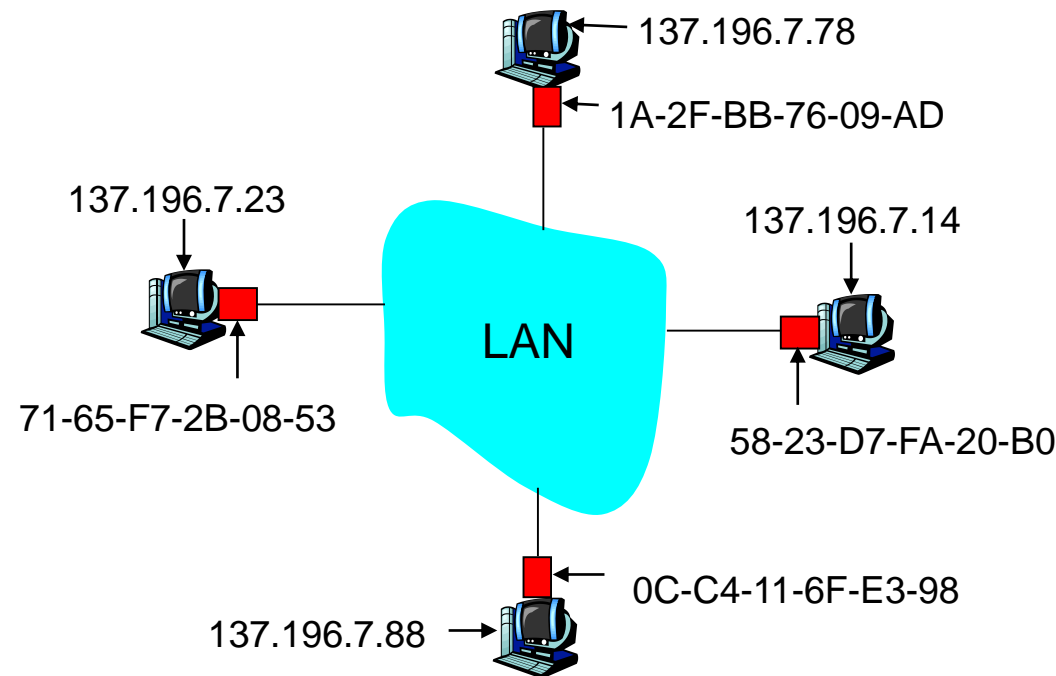
# ARP: Address Resolution Protocol

*Question:* how to determine MAC address of B knowing B's IP address?

- Each IP node (host, router) on LAN has **ARP** table
- ARP table: IP/MAC address mappings for some LAN nodes

< IP address; MAC address; TTL >

- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

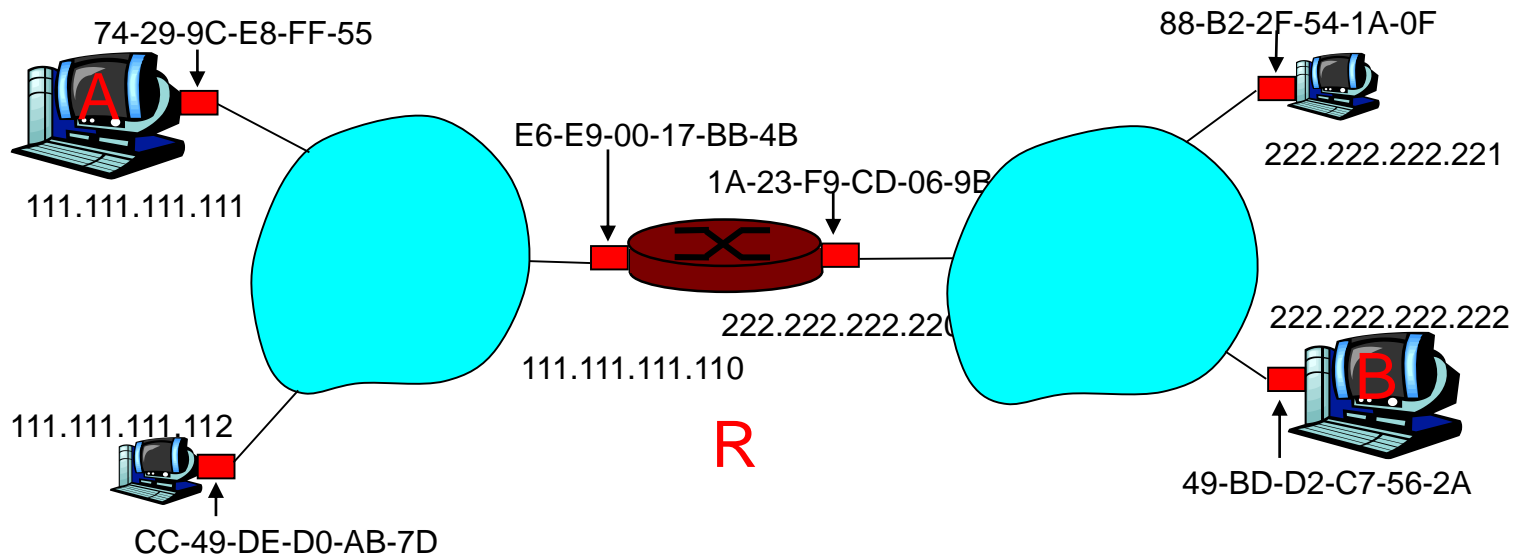


# ARP protocol: Same LAN (network)

- A wants to send datagram to B, and B's MAC address not in A's ARP table.
- A **broadcasts** ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is “plug-and-play”:
  - nodes create their ARP tables *without intervention from net administrator*

# Addressing: routing to another LAN

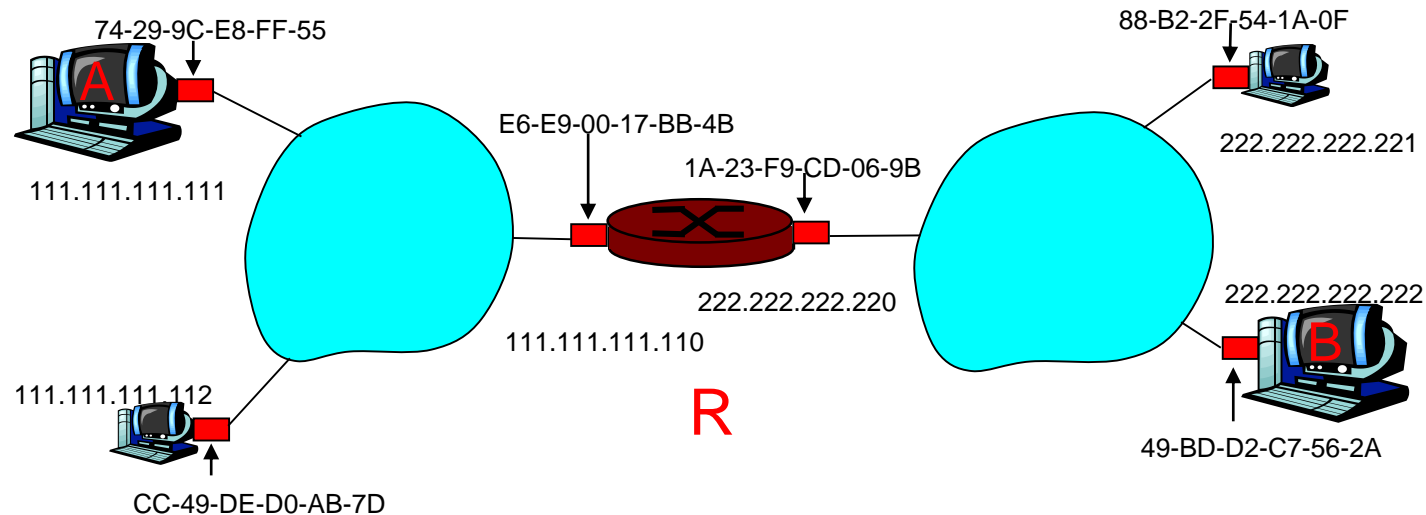
walkthrough: send datagram from A to B via R  
assume A knows B's IP address



- two ARP tables in router R, one for each IP network (LAN)

- A creates IP datagram with source A, destination B
- A uses ARP to get R's MAC address for 111.111.111.110
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram
- A's NIC sends frame
- R's NIC receives frame
- R removes IP datagram from Ethernet frame, sees its destined to B
- R uses ARP to get B's MAC address
- R creates frame containing A-to-B IP datagram and sends to B

This is a **really** important example – make sure you understand!



# Link Layer

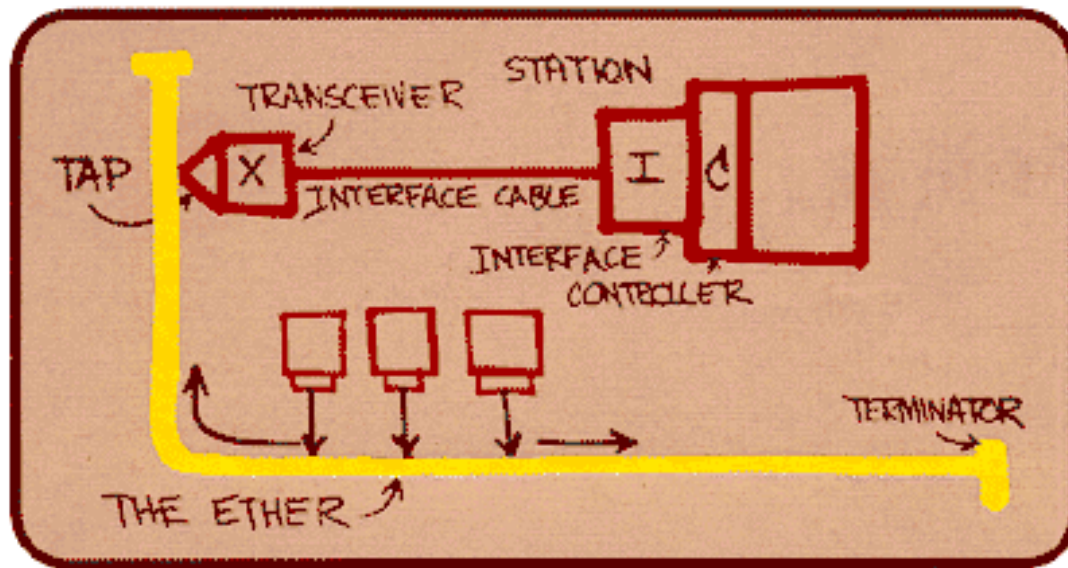
- 2.1 Introduction and services
- 2.2 Error detection and correction
- 2.3 Multiple access protocols
- 2.4 Link-Layer Addressing
- 2.5 Ethernet
- 2.6 Link-layer switches
- 2.7 PPP
- 2.8 Link Virtualization: ATM and MPLS



# Ethernet

“dominant” wired LAN technology:

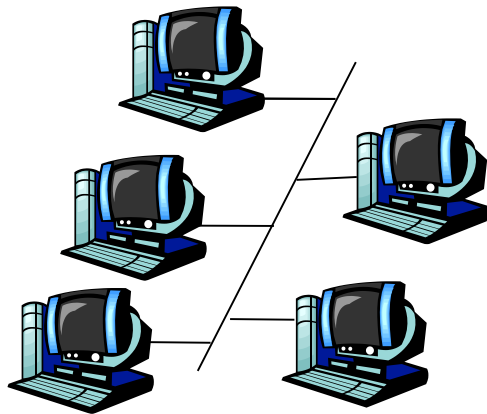
- cheap \$20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 40/100 Gbps



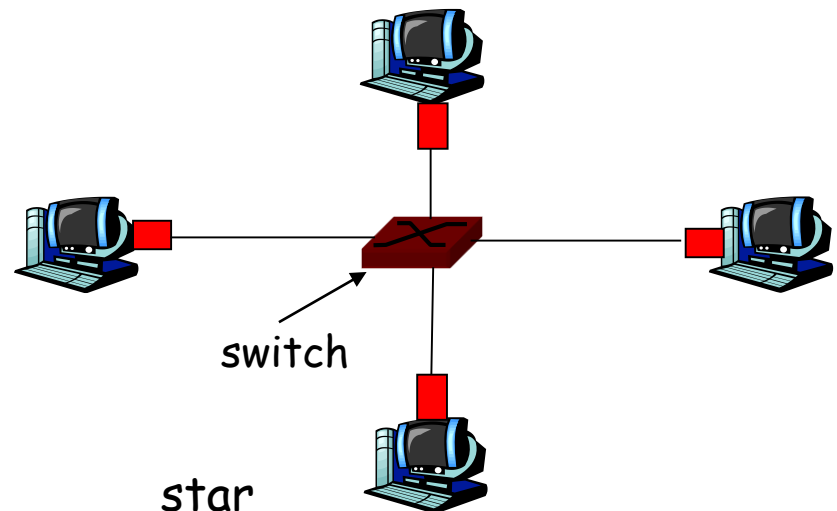
Metcalfe's Ethernet sketch

# Star topology

- bus topology popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- today: star topology prevails
  - active *switch* in center
  - each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)



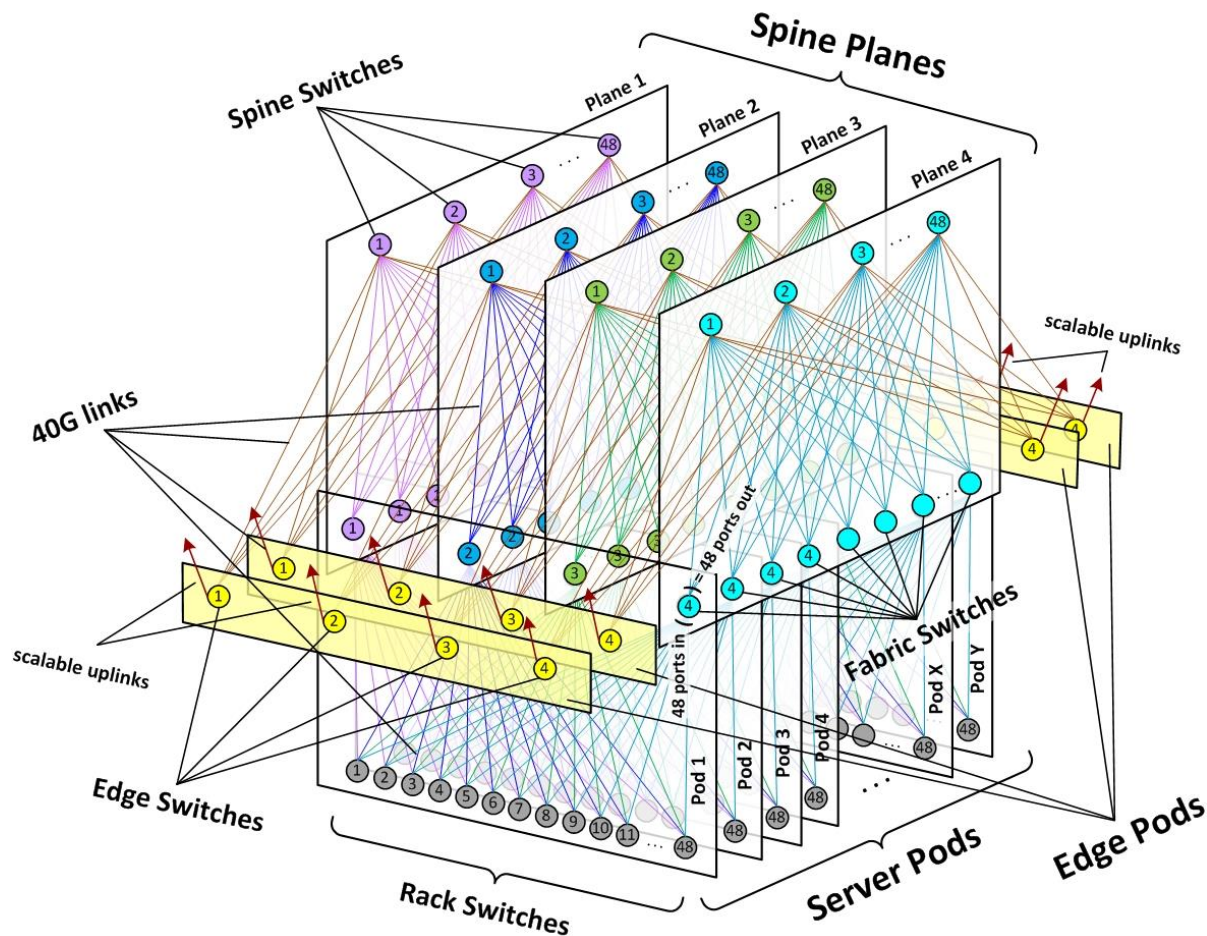
bus: coaxial cable



star

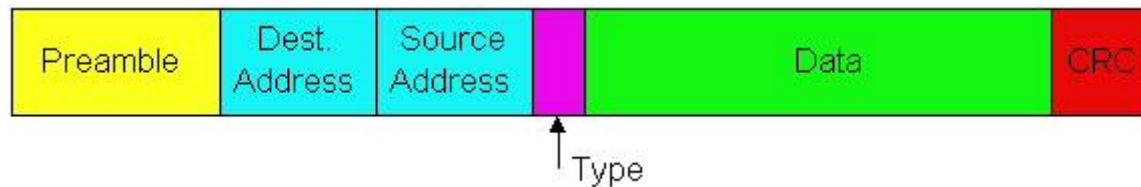
# Datacenter topology

- Hierarchical tree topology
  - Multiple layers of switches
  - Possibly tens of thousands of servers



# Ethernet Frame Structure

Sending adapter encapsulates IP datagram (or other network layer protocol packet) in **Ethernet frame**

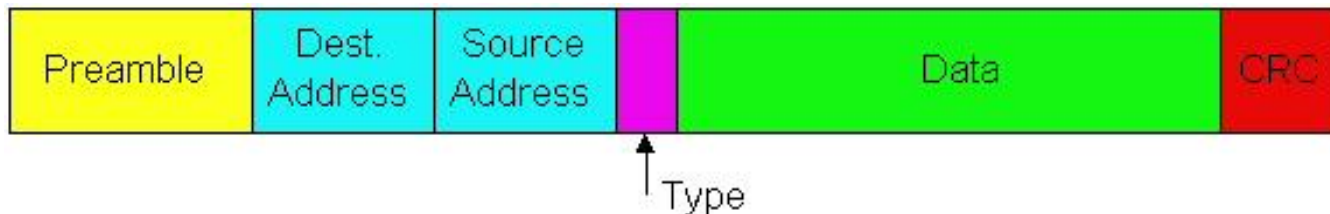


## Preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

# Ethernet Frame Structure (more)

- **Addresses:** 6 bytes
  - if adapter receives frame with matching destination address, or with broadcast address (eg ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- **Type:** indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- **CRC:** checked at receiver, if error is detected, **frame is dropped**



# Ethernet: Unreliable, connectionless

- **connectionless**: No handshaking between sending and receiving NICs
- **unreliable**: receiving NIC doesn't send acks or nacks to sending NIC
  - stream of datagrams passed to network layer can have gaps (missing datagrams)
  - gaps will be filled if app is using TCP
  - otherwise, app will see gaps
- Ethernet's MAC protocol: unslotted **CSMA/CD**

# Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission  
If NIC senses channel busy, waits until channel idle, then transmits
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame !
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters **exponential backoff**: after  $m$ th collision, NIC chooses  $K$  at random from  $\{0, 1, 2, \dots, 2^m - 1\}$ . NIC waits  $K \cdot 512$  bit times, returns to Step 2

# Ethernet's CSMA/CD (more)

**Jam Signal:** make sure all other transmitters are aware of collision; 48 bits

**Bit time:** .1 microsec for 10 Mbps Ethernet ;  
for  $K=1023$ , wait time is about 50 msec

**Exponential Backoff:**

- *Goal:* adapt retransmission attempts to estimated current load
  - heavy load: random wait will be longer
- first collision: choose  $K$  from  $\{0,1\}$ ; delay is  $K \cdot 512$  bit transmission times
- after second collision: choose  $K$  from  $\{0,1,2,3\}$ ...
- after ten collisions, choose  $K$  from  $\{0,1,2,3,4,\dots,1023\}$



# CSMA/CD efficiency

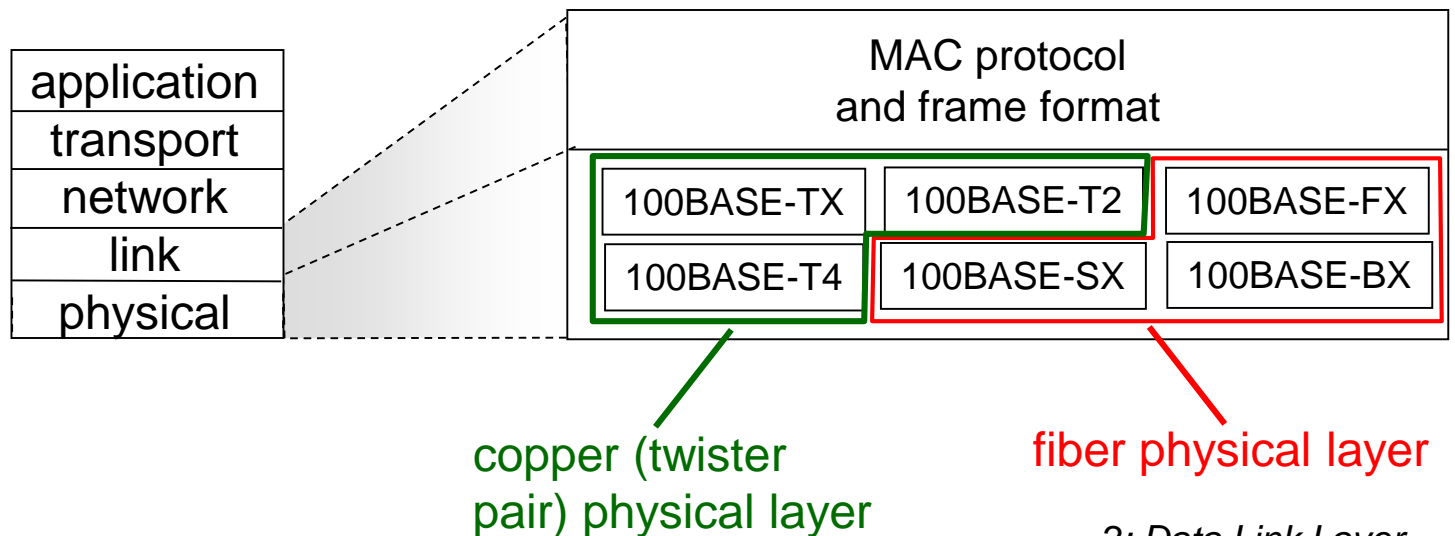
- $t_{prop}$  = max prop delay betw. 2 nodes in LAN
- $t_{trans}$  = time to transmit max-size frame

$$efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

- efficiency goes to 1
  - as  $t_{prop}$  goes to 0
  - as  $t_{trans}$  goes to infinity
- better performance than ALOHA: and simple, cheap, decentralized!

# 802.3 Ethernet Standards: Link & Physical Layers

- *many* different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps, ...
  - different physical layer media: fiber, cable



# Link Layer

- 2.1 Introduction and services
- 2.2 Error detection and correction
- 2.3 Multiple access protocols
- 2.4 Link-layer Addressing
- 2.5 Ethernet
- 2.6 Link-layer switches (next lecture)
- 2.7 PPP
- 2.8 Link Virtualization: ATM, MPLS